Don't forget that this is a Paired Programming assignment, so you should continue to work with your partner on this. If you have not yet switched roles between 'driver' and 'navigator', be sure to do so now. Remember to continue to switch roles early and often for this entire project.

Background: `Block`

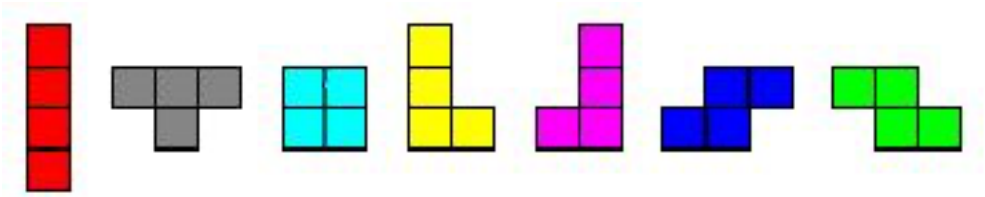
We will be using the `Block` class you created in Part 0 of this project. Each instance of the `Block`class will represent a single 1x1 block in the game of Tetris. For your reference, below is a summary of its constructor and methods:

`Block` class

```
Color getColor()
void setColor(Color newColor)
Grid getGrid()
Location getLocation()
void putSelfInGrid(Grid grid, Location location)
void removeSelfFromGrid()
void moveTo(Location newLocation)
String toString()
```

Exercise 1.0: The Communist Bloc

Shapes composed of four blocks each are called *tetrads*, the leading actors of the Tetris world. Tetrads come in seven varieties: I*, T, O, L, J, S,* and *Z*. They are shown here.



Open the `Tetrad` class, which keeps track of an array of four `Block` objects. (Note that the grid only keeps track of the blocks and knows nothing about tetrads. Instead the `Tetris` class will eventually keep track of the tetrad being dropped.)

Recall that a `Block` is an object that keeps track of its color, location, and the `Grid` that contains it, all of which are appropriately managed by calling its `putSelfInGrid` method. Go ahead and complete the private helper method in the `Tetrad` class called `addToLocations`, which adds each of the four blocks (contained in the instance variable array `blocks`) to the four locations (in the parameter array `locations`) in the given `grid`, according to the following.

```
// precondition:  blocks are not in any grid;
//                blocks.length = locations.length = 4.
// postcondition: The locations of blocks match locations,
//                and blocks have been put in the grid.
private void addToLocations(Grid grid, Location[] locations)
```

Run `TetradTest` to check that you have completed this exercise before moving on.

Exercise 1.1: I, Tetrad

Now, look at the constructor, which takes in the grid and initializes this `Tetrad` as one of the seven tetrad shapes. Here is an outline of how to complete the constructor.

- First, initialize the instance variable array of `Block` objects to contain four new `Block` objects. There are two steps here; first, instantiate the array to a length of four, and then loop through the array and initialize (load) each position in the array with a new `Block` object.

Run `TetradTest` to check that you have completed this exercise before moving on.

Exercise 1.2: Four Blocks and Seven Shapes Ago, ...

- Modify the `Tetrad` class's constructor so that it first picks a random shape number from 0 to 6. If it picks 0, it will create a red I-shaped tetrad as before. However, for each of the other numbers, it will pick one of the other 6 shapes.

Run `TetradTest` to check that you have completed this exercise before moving on.

Exercise 1.3: Block Party

Continuing in the constructor, note that all tetrad shapes have already been defined for you, including their color. Feel free to change the color if you wish, but every shape should have a unique color. The `Color` class contains predetermined color constants (BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW), or you can create your very own colors if you wish.

- Next, at the bottom of the constructor (after the if statements), set the color of each `Block` to be the color indicated by the `color` variable.
- Lastly, have the constructor call `addToLocations` to add the blocks using the array of `Location` objects.

Run `TetradTest` to check that you have completed this exercise before moving on. Please submit a screenshot of the tester success message for Part - 1.